The Research and Development of ChemGrid in CGSP

Ying Zhao[1], Ye Yue[1], Xuelin Shi[1], Jianwei Li[1], Atul Sajjanhar[2]
[1] *School of Information Science and Technology, Beijing University of Chemical Technology, Beijing, 100029, P.R.China*
[2] *School of Engineering and Information Technology, Deakin University, 221 Burwood HWY, Burwood, VIC 3125, Australia*
{zhaoy, yueye, shixl, lijw}@mail.buct.edu.cn; atuls@deakin.edu.au

Abstract

With the rapid development of computing technologies and network technologies, Grid technology has emerged as the solution for high-performance computing. Recently, the grid of orient-services has become a hot issue in this research area. In this paper, we propose an architecture of ChemGrid in CGSP (China Grid Support Platform). The effectiveness of the proposed architecture is demonstrated by an example which is developed as a Web service based on CGSP; the Web service is used for searching elements in the periodic table. An improvement of the user interface for applications is proposed in order to obtain results interactively. Finally, an extension of ChemGrid is discussed in order to integrate different types of resources and provide specialized services.

1. Introduction

Chemical engineering involves the design and manufacturing of new chemical compounds and materials, the analysis of chemical reactions, the simulation of computational fluid dynamics (CFD), the simulation of molecular movement, and so on. To lower costs and improve efficiency, information technology (IT) is increasingly employed in chemical engineering [1]. One of key IT component is grid computing, which can construct a virtual single image of heterogeneous resources, provide uniform application interface and integrate widespread computational resources into super, ubiquitous and transparent aggregation [2].

CGSP is a grid middleware in service-oriented architecture developed for China Education and Scientific Research Grid Project. This paper first discusses the implementation methods of ChemGrid in CGSP and gives an example of an application which enables searching of the periodic table of elements. Some applications need to get results quickly and interactively, so we propose an improvement of user interfaces provided by CGSP2.0. Finally, we use Web Services and Agent technologies to extend the current architecture of ChemGrid.

The paper is organized as follows: In Section 2, architecture of ChemGrid is built based on CGSP. Section 3 introduces the implementation methods of Web Services with access to resources in CGSP. An improvement of the user interface is presented in Section 4. Section 5 discusses an extension of the current ChemGrid. The conclusion is made at the end of paper.

2. ChemGrid Framework Based On CGSP

The ChemGrid is a chemical grid built by Beijing University of Chemical Technology. It is supported by China Grid Support Platform (CGSP), and is also a grid computing application platform for chemical engineering applications. The details are described below.

2.1. China Grid Support Platform (CGSP)

CGSP is a grid middleware developed to build the ChinaGrid, which integrates all kinds of resources in education and research environments, makes the heterogeneous and dynamic nature of resources transparent to the users, and provides high performance, high reliability, secure, convenient and transparent grid service for scientific computing and engineering research. CGSP provides both ChinaGrid service portal, and a development environment for deploying various grid applications.

2.2. Architecture of ChemGrid

978-0-7695-3306-3/08 $25.00 © 2008 IEEE
DOI 10.1109/ChinaGrid.2008.15

The system of Chemical Grid provides a web interface for users to access the above services. Users can select the service they need and submit jobs. Then the system invokes the service to execute the jobs. The results can be acquired from a webpage or a file system. Fig.1 shows the system architecture of Chemical Grid:

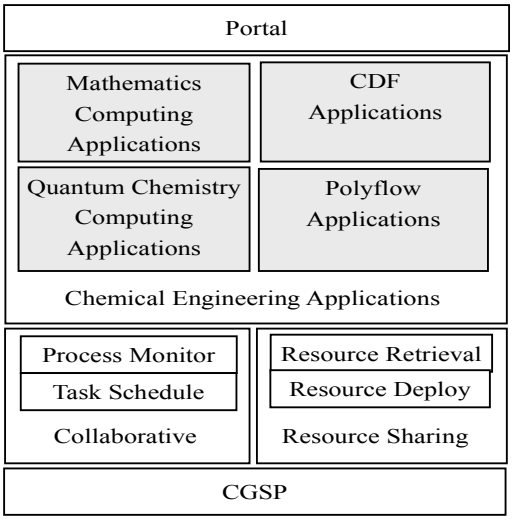

Figure 1. Architecture of ChemGrid

The above applications can cover most of the requirements of chemical engineering researchers in Beijing University of Chemical Technology (BUCT), China. By defining standard workflow and general interfaces for every type of application, the platform facilitates large-scale computer-related resources and services for users. Therefore all these users can access any service on Chemical Grid platform and share the resources fairly.

The portal is a web interface for users, which is the entry point for the end user to use grid services. By using the portal, users can submit their jobs, monitor the execution of jobs, manage and transfer data, inquiry the grid resource information.

Collaborative environment can also provide job scheduling and monitoring functions. When receiving a user's request, it locates a suitable node in the grid to perform the computation. Resource sharing environment provides deeper sharing and management capability for domain specific resources.

A scenario of an example job execution workflow is shown in Fig.2. Firstly, user inputs the computing requests according to the submission form from the web page generated by the portal of ChemGrid platform.

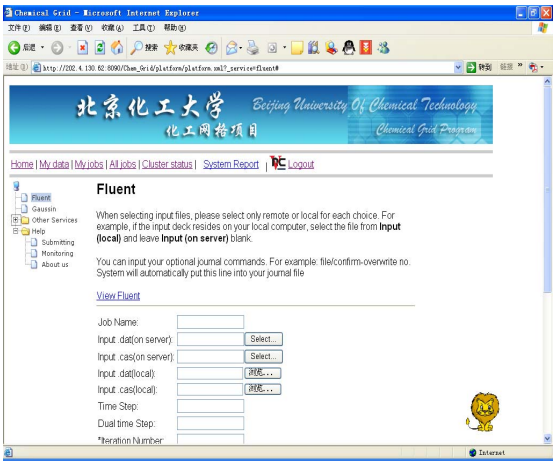

Figure 2. Web Interface of ChemGrid Platform

The ChemGrid is a chemical grid built by Beijing University of Chemical Technology. It is supported by China Grid Support Platform (CGSP), and is also a grid computing application platform for chemical engineering applications. The details are described below.

3. Grid Services Based on CGSP

CGSP is based on the core of Globus Toolkit 3, and is compatible with WSRF and OGSA. CGSP 2.0 provides a powerful platform to execute various jobs, including legacy programs, Web services, WS-Resources and Composite Services. All the external components are defined in WSDL, which is derived from Globus Services directly. When multiple resources are involved, Web Services Resource Frame should follow factory/instance module and Web resources factory module. Factory service is responsible for creating resources, and Instance service is used for accessing information of a resource. In order to explain services clearly, the paper presents an example of an application, namely, Periodic Table of Elements which demonstrates the implementation of Web Services deployed in CGSP2.0. The users can provide parameters like atomic weight, discovery day, or discoverer to this service and can get all the information about the element. The six steps describing this Web Service are as follows.

3.1. Define the interface in WSDL

The first step of creating a Web service is to define an interface of the service, which does not provide low-level details about implementation, just like an implementation algorithm. The service and code are separated and only operations are defined in this step.

The syntax of WSDL follows XML format. The definition code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
< definitions name="Elements"
targetNamespace=http://www.chinagrid.edu.cn/nam
 espaces/test/elem/Elements_instance
...
<types>
targetNamespace="http://www.chinagrid.edu.cn/na
 mespaces/test/elem/Elements_instance"
xmlns:tns=http://www.chinagrid.edu.cn/namespaces
 /test/elem/Elements_instance
...
<portType name="ElementsPortType"
 wsdlpp:extends="wsrpw:GetResourceProperty"
wsrp:ResourceProperties="tns:ElementsResourcePr
 operties">
...
```

The content ("wsrpw:GetResourceProperty) in label "< portType >" is not standard WSDL, but a part of namespace (WSDLPreprocessor) provided by Globus. This reference can tell us that WSDL Preprocessor contains "GetResourceProperty portType" from WS-ResourceProperties [5]. WSDL Preprocessor is provided by GT4 which specifies some resource properties, and is described as:

"tns:ElementsResourceProperties"

3.2. Implement service in Java

After the definition of the Web service, the next step is to implement the service in Java. However, if this program needs to access MySQL database and read Chinese characters from the database, it is necessary to specify the encoding type as "UTF-8".

```
public interface ElementsQNames {
 public static final String NS =
 "http://www.chinagrid.edu.cn/namespaces/test/ele
 m/Elements_instance";
   public static final QName RP_STATUS = new
 QName(NS, "Status");
 public static final QName
 RESOURCE_PROPERTIES = new QName(NS,
 "ElementsResourceProperties");}
private ResourcePropertySet propSet;
private String status;
...
this.propSet = new
 SimpleResourcePropertySet(ElementsQNames.RES
 OURCE_PROPERTIES);
status="";
```

3.3. Create WSDD and JNDI configuration file

Up to this point, we have written the two most important parts of our stateful Web service: the service interface (WSDL) and the service implementation (Java). How do we make the web service available to client requests? In this step, we will take all the loose pieces we have written up to this point and make them available through a Web services container. The WSDD (Web Service Deployment Descriptor) is used for source mapping and the JNDI (Java Naming and Directory Interface) is responsible for multiple parameters and resources. The following is WSDD file where the service name should be set "test/elem/Elements" for the application call.

```
...
<service name="test/elem/Elements"
 provider="Handler" use="literal"
 style="document">
<parameter name="className"
 value="cn.edu.chinagrid.test.elem.service.impl.Ele
 ments"/>
...
```

Because our service only use one resourse, JNDI configuration is definied simplely：

```
<?xml version="1.0" encoding="UTF-8"?>
<jndiConfig
 xmlns="http://wsrf.globus.org/jndi/config">
<service name="test/elem/Elements">
```

3.4. Create GAR files

This GAR file is a single file which contains all the files and information the Web services container needs to deploy our service and make it available to the whole world. The services generator can finish generalRunningservice style file automatically. First, there should be a folder. Second, the build.xml (provided by Globus), Java code file, WSDD and JNDI file are put into this folder. Then it is time to create the GAR file. The following is the command line to create the GAR file.

```
./globus-build-service.sh -d <service base
directory> -s <services WSDL file>
```

"globus-build-service.sh"（ globus-build-service.py in Windows）is provided by Globus to create the GAR file. In here, "<service base directory>" is the directory of the Java code file and "<services WSDL file>" is the directory of the WSDL file. It should be noted that this service needs an activated file, if not, CGSP can not find where the service is. This file is also WSDL file format.

```
<parameter id="getElem">
 <caption>getElem</caption>
...
```

<operation name="getElem">
<soapAction>
http://www.chinagrid.edu.cn/workflow_job_sample/image_processing/
</soapAction>
...
</operation>

3.5. Deploy the service into a Web Services container

The GAR file contains all the files and information the web server needs to deploy the web service [6]. Deployment is done with a GlobusToolKit4 tool "globus-deploy-gar", which unpacks the GAR file into key locations in the CGSP directory tree. The command line is shown as follows:

$GLOBUS_LOCATION/bin/globus-deploy-gar ./elem.gar

The "elem.gar" is from the previous step. On error-free completion of the operation, the "successful" sign will appear.

3.6. Register service

The service should be registered on a node of CGSP. The main purpose is to allow users to access services on the portal. One grid platform may include a lot of nodes, all unregistered services could not be executed even if these services were deployed. These services must be registered to some nodes for execution. The registry information is given as following:

Name: C-service
Wsdl: http://222.199.242.21:9090/wsrf/services/test/elem/ElementsService/wsdl
Desc: this is an Elements-Search service
Address: http://222.199.242.21/wsrf/services/test/elem/ElementsService
Keyword: Elements-Search
Catalog: Chemistry
Invoke Doc: http://222.199.242.21:9090/share/schema/elem/Elements_desc.wsdl
NRS Address: http://222.199.242.21:9090/wsrf/services/NodeRegistryService

4. Improvement of user interface

CGSP is based on the core of Globus Toolkit 3 , and is compatible with WSRF and OGSA. CGSP 2.0 provides a powerful platform to execute various jobs, including legacy programs, Web Services, WS-Resource and Composite Service which is actually a defined workflow. Furthermore in order to provide a user friendly interface to utilize CGSP functions, CGSP portal also provides webpage interface for end users. From the web portal, users can browse services and resources in the grid, view users' data space, upload and download files with http or gridftp, submit jobs to applications and services.

The services deployed on CGSP are unfriendly because the user can not get results immediately after the job is run successfully. If the user wants to get results quickly, he or she must submit another job to get the results, which is unfriendly. To solve the problem, it is necessary to know the execution process of Web services. CPDK is a Globus development tool based on Globus and Java CoG. It uses JavaBean/JSP to implement the portal in Tomcat. The important function of CPDK is to initialize the portal engine, which contains some basic portal information, like log, job submit, job supervise and the portal information database. The portal also provides an authentication certificate of authorization and user management. When the Web explorer sends an http/https-request to the portal server, CPDK wakes up the page. Page look-up table is a configuration file of object mapping active pages. An active page does a logic Grid portal operation and call the service program to perform the requested operation. In the end, it transfers the control to a view page which is generated by a JSP Servlet.

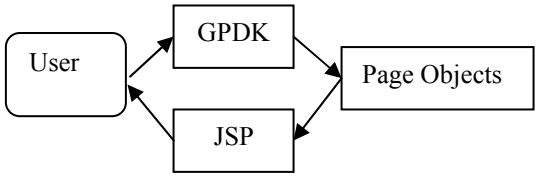

Figure 3. The process of work

Since then the JSP Servlet should get the result of job. Some code from CGSP2.0 portal is described below.

```
HashMap para=new HashMap();
para.put("instanceID",s);
JobInstanceInfo
  info=jobmonitor.querySingleJob(para);
```

The expected result is in info.getJobResult(). However, besides the result, there are some job labels like job ID. So we make a program to filter the labels and retain the result that the user needs. Fig. 4 shows the results:

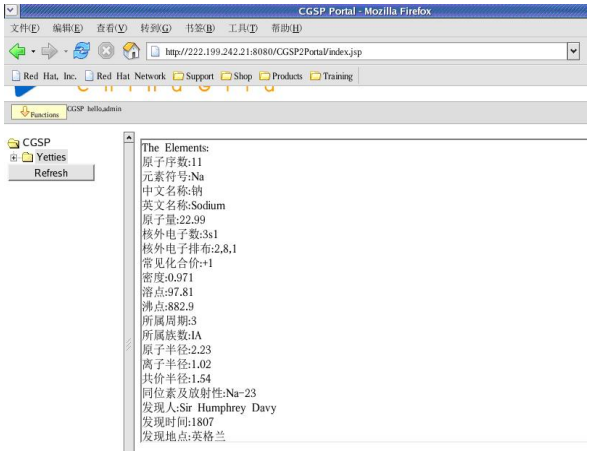

Figure 4. The result of portal improvement

The core code snippet is described as follows:

```
ServiceInvoke
sk=(ServiceInvoke)session.getAttribute(JobSessio
n.SERVICEINVOKE);
```

Where sk is a ServiceInvoke type which has been defined in the background and gets the initial information of the job. Besides it also needs a CGSPAdapter:

```
CGSPAdapter cgsp = (CGSPAdapter)
session.getAttribute("cgsp");
```

Then the function "sk.submitNoBlock(cgsp.getJobSubmitter(), request, session)" can do the job. If all of the above is successful, the result will appear.

Furthermore, the results are in the info.getJobResult(). Besides the result, there are some job labels, just like job ID. So we make a C program to filtrate this labels and keep back the result that user want. The main code includes:

```
void analyzeXML(string &Content){
  if ('<'==xmlContent[0])
  {
    size_t it= Findfirst(Content,'>');
    string tagName = Substr(1,it-1,Content);
        ...
    //Find end tag
    string tagEndStr= "</"+nodeName+">";
   size_t tagIndex=find(Content,tagEndStr);
        ...
    //Get the content between two tag
   string tagMid=Substr(++it,tagIndex-it,Content);
   analyzeXML(tagMid); // Recursive
 }
```

5. The Extension of ChemGrid

We have built the ChemGrid project to provide large-scale chemical computing, however, some resources are very private and some software are computer sensitive. It is very difficult to integrate these resources into our ChemGrid. Actually, we have been working in Web services and have built some services for chemical engineering applications. Web services are defined as self-contained, modular units of application logic which provide business functionality to other applications via an Internet connection. So we can integrate these pure Web services into ChemGrid, and extend the services and functions of ChemGrid.

Web Services as an emerging technology has good prospects for development. Normally it includes three roles: service broker, service provider, and service requestor. The Web services platform is usually perceived as a combination of XML, HTTP, SOAP, WSDL, UDDI. The new architecture of new ChemGrid is shown in Fig. 5.

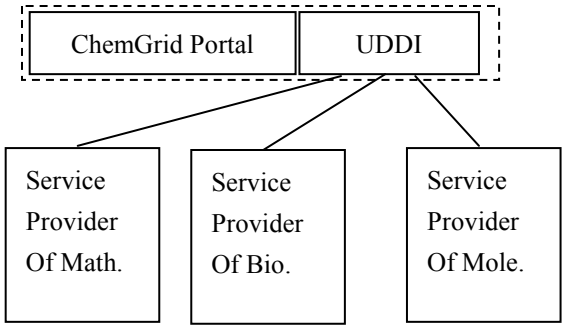

Figure 5. Extension of ChemGrid

In this architecture, a UDDI is integrated into the portal of ChemGrid. This UDDI registry is similar to a CORBA trader, or it can be thought of as a DNS service for business applications. It has two kinds of clients: service provider that wants to publish a service (and its usage interfaces), and users who want to obtain services of a certain kind. In here, Apache JUDDI is applied to ChemGrid.

Providers of Web services are generally known as application service providers. Some applications software belong to an organization, and they hope that they are self-manageable. The owner of applications can control when they can be used, and who can use them. When a service is created and pushed to UDDI registry, the portal of ChemGrid will show this service to users. When the service owner does not want to provide service, it can revoke this service from UDDI. The portal will delete this service immediately.

Until now we have been working to use CGSP to register these services, but we failed to achieve a seamless connection. An independent UDDI register had to be provided in order to publish services quickly.

6. Conclusion and future work

In this paper, we present an architecture of ChemGrid, developed by BUCT, which supports chemical engineering applications related to computation, simulation, and virtualization. Since CGSP considers all the resource as grid services, developing and deploying a service on CGSP is sensible. The example of periodic table of elements demonstrates the steps and methods in creating a Web Service for ChemGrid.

When the user wants to get results from ChemGrid, the user needs to enter their user space to retrieve results. The user is unable to get results on a Web page. In order to provide a user friendly interface to utilize CGSP functions, we make an improvement of CGSP portal and can provide results on a Web page immediately.

Finally, an extension of ChemGrid by Web Services is discussed. The owner of resources can create its own Web Services, and publish them to the portal of ChemGrid, which has integrated Apache JUDDI. The end user can access the services from this portal. This means a lot of resources can be joined into ChemGrid by Web services technology.

In the future, Chemical Grid will integrate more chemical engineering applications and resources.

Acknowledgements

This paper has been supported by the Chemical Grid Project of Beijing University of Chemical Technology.

References

[1] Ying Zhao, Xuelin Shi, “Collaborative Computational Chemical Grid Based on CGSP”, *Proceedings of NPC 2007*, Oct. 2007, pp.199-202.

[2] Jin Hai, “China Grid: making grid computing a reality”, *Proceedings of ICADL’04, Lecture Notes in Computer Science*, Vol.3334,2004,pp.13-24.

[3] Ian Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, *U.S.A.:Math & Computer Science Division*, Argonne National Lab，2005.

[4] Novotny J., “The Grid Portal Development Kit”, *Concurrency and Computation: Practice and Experience*, 2002,14(13), pp. 1128-1145.

[5] Foster I, Czajkowski K, Fcrguson D F., “Modeling and Managing state in Distributed systems:The Role of OGSI and WSRF”, *proccedings of the IEEE,* 2005,93(3), pp.603-613.

[6] Borja Sotomayor, “The Globus Toolkit 4 Programmer's Tutorial”, *http://gdp.globus.org/gt4-tutorial*, 2005.